实验编号：2**四川师大 编译原理 实验报告 2020**年**4**月**12**日

**计算机科学学院** 2017级 4 班 实验名称： LL(1)分析法

姓名：\_韩勇\_\_\_ 学号：\_2017110409\_ 指导老师：\_\_吴贞东\_\_ 实验成绩:\_\_\_\_\_

**实验 二 LL(1)分析法\_**

1. 实验目的及要求

通过完成预测分析法的语法分析程序，了解预测分析法和递归子程序法的区 别和联系。使学生了解语法分析的功能，掌握语法分析程序设计的原理和构造方法， 训练学生掌握开发应用程序的基本方法。有利于提高学生的专业素质，为培养适应 社会多方面需要的能力。

1、编程时注意编程风格：空行的使用、注释的使用、缩进的使用等。

2、如果遇到错误的表达式，应输出错误提示信息。

3、对下列文法，用LL（1）分析法对任意输入的符号串进行分析： （1）E->TG （2）G->+TG|—TG （3）G->ε （4）T->FS （5）S->\*FS|/FS （6）S->ε （7）F->(E) （8）F->i 输出的格式如下：

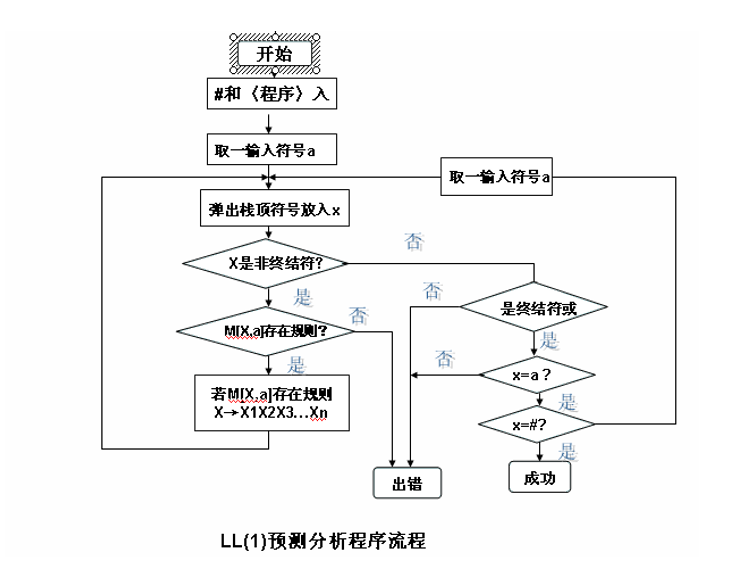


1. 实验内容

根据某一文法编制调试LL（1 ）分析程序，以便对任意输入的符号串进 行分析。 构造预测分析表，并利用分析表和一个栈来实现对上述程序设计语言的分析 程序。 分析法的功能是利用LL（1）控制程序根据显示栈栈顶内容、向前看符号 以及LL（1）分析表，对输入符号串自上而下的分析过程。

1. 实验主要流程、基本操作或核心代码、算法片段（该部分如不够填写，请另加附页）

流程图：



源程序：

#include<iostream>

#include<stack>

#include<string>

using namespace std;

string Production[10]={"E->TG","G->+TG","G->-TG","G->ε","T->FS","S->\*FS","S->/FS","S->ε","F->(E)","F->i"};//产生式集合

char termina[8]={'+','-','\*','/','(',')','i','#'};//终结符集合

char nontermina[5]={'E','T','G','F','S'};//非终结符集合

int count=0;//记录当前进行处理的输入字符串字符位置

int line=0;//记录处理的步骤数

string stacktd="#E";//记录栈中内容

stack <char> Stack;//创建一个栈

char ch,ch1;//ch记录栈顶字符，ch1记录当前进行处理的输入字符串字符

int Istermina(char c){//判断其是否为终结符

for(int i=0;i<8;i++){

if(termina[i]==c)

return 1;

}

return 0;

}

int Isnontermina(char c){//判断其是否为非终结符

for(int i=0;i<5;i++){

if(nontermina[i]==c)

return 1;

}

return 0;

}

void OutputString(string s){//输出未处理的字符串

for(int i=count;i<s.size();i++)

cout<<s.at(i);

}

void Analyse(string s){//分析过程

s=s+"#";

Stack.push('#');//初始化

Stack.push('E');

cout<<line<<"\t"<<stacktd<<"\t";

OutputString(s);

cout<<"\t\t"<<"\t\t"<<"初始化"<<endl;

line++;

while(count<s.size()){//循环条件，输入串是否被处理完全

ch=Stack.top();

ch1=s.at(count);

if(Istermina(ch)){

if(ch==ch1&&ch!='#'){

stacktd.erase(stacktd.length()-1);//删除stacktd字符串的尾字母

Stack.pop();

count++;

cout<<line<<"\t"<<stacktd<<"\t";

OutputString(s);

cout<<"\t\t"<<"\t\t"<<"GETNEXT("<<ch<<")"<<endl;

}

else if(ch='#'){

cout<<"acc"<<"\t分析成功"<<endl;//表示分析完成，输入串为该文法的正确句型

count=s.size();

}

else{

cout<<"\tError"<<endl;//表示输入串为该文法的非法句型

count=s.size();//使循环结束

}

}

else switch(ch){//switch语句，具体分析非终结符面对输入串的具体执行情况

case'E':{if(ch1=='('||ch1=='i'){

stacktd.erase(stacktd.length()-1);

Stack.pop();

Stack.push('G');

Stack.push('T');

stacktd=stacktd+"GT";//将输入栈中的内容加到stacktd中

cout<<line<<"\t"<<stacktd<<"\t";

OutputString(s);

cout<<"\t\t"<<"E->TG"<<"\t\t"<<"POP,PUSH(GT)"<<endl;

}

else{

cout<<"\tError"<<endl;

count=s.size();

}

}break;

case'T' :{if(ch1=='('||ch1=='i'){

stacktd.erase(stacktd.length()-1);

Stack.pop();

Stack.push('S');

Stack.push('F');

stacktd=stacktd+"SF";

cout<<line<<"\t"<<stacktd<<"\t";

OutputString(s);

cout<<"\t\t"<<"T->FS"<<"\t\t"<<"POP,PUSH(SF)"<<endl;

}

else{

cout<<"\tError"<<endl;

count=s.size();

}

}break;

case'G' :{if(ch1=='+'){

stacktd.erase(stacktd.length()-1);

Stack.pop();

Stack.push('G');

Stack.push('T');

Stack.push('+');

stacktd=stacktd+"GT+";

cout<<line<<"\t"<<stacktd<<"\t";

OutputString(s);

cout<<"\t\t"<<"G->+TG"<<"\t\t"<<"POP,PUSH(GT+)"<<endl;

}

else if(ch1=='-'){

stacktd.erase(stacktd.length()-1);

Stack.pop();

Stack.push('G');

Stack.push('T');

Stack.push('-');

stacktd=stacktd+"GT-";

cout<<line<<"\t"<<stacktd<<"\t";

OutputString(s);

cout<<"\t\t"<<"G->-TG"<<"\t\t"<<"POP,PUSH(GT-)"<<endl;

}

else if(ch1==')'||ch1=='#'){

stacktd.erase(stacktd.length()-1);

Stack.pop();

cout<<line<<"\t"<<stacktd<<"\t";

OutputString(s);

cout<<"\t\t"<<"G->ε"<<"\t\t"<<"POP"<<endl;

}

else{

cout<<"\tError"<<endl;

count=s.size();

}

}break;

case'F' :{if(ch1=='('){

stacktd.erase(stacktd.length()-1);

Stack.pop();

Stack.push(')');

Stack.push('E');

Stack.push('(');

stacktd=stacktd+")E(";

cout<<line<<"\t"<<stacktd<<"\t";

OutputString(s);

cout<<"\t\t"<<"F->(E)"<<"\t\t"<<"POP,PUSH()E()"<<endl;

//cout<<")E("<<endl;

}

else if(ch1=='i'){

stacktd.erase(stacktd.length()-1);

Stack.pop();

Stack.push('i');

stacktd=stacktd+"i";

cout<<line<<"\t"<<stacktd<<"\t";

OutputString(s);

cout<<"\t\t"<<"F->i"<<"\t\t"<<"POP,PUSH(i)"<<endl;

}

else{

cout<<"\tError"<<endl;

count=s.size();

}

}break;

case'S' :{if(ch1=='\*'){

stacktd.erase(stacktd.length()-1);

Stack.pop();

Stack.push('S');

Stack.push('F');

Stack.push('\*');

stacktd=stacktd+"SF\*";

cout<<line<<"\t"<<stacktd<<"\t";

OutputString(s);

cout<<"\t\t"<<"S->\*FS"<<"\t\t"<<"POP,PUSH(SF\*)"<<endl;

}

else if(ch1=='/'){

stacktd.erase(stacktd.length()-1);

Stack.pop();

Stack.push('S');

Stack.push('F');

Stack.push('/');

stacktd=stacktd+"SF/";

cout<<line<<"\t"<<stacktd<<"\t";

OutputString(s);

cout<<"\t\t"<<"S->/FS"<<"\t\t"<<"POP,PUSH(SF/)"<<endl;

}

else if(ch1=='+'||ch1=='#'||ch1=='-'||ch1==')'){

stacktd.erase(stacktd.length()-1);

Stack.pop();

cout<<line<<"\t"<<stacktd<<"\t";

OutputString(s);

cout<<"\t\t"<<"S->ε"<<"\t\t"<<"POP"<<endl;

}

else{

cout<<"\tError"<<endl;

count=s.size();

}

}break;

default : {cout<<ch<<"\tError"<<endl;

count=s.size();

}

}

line++;

}

}

int main(){

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*LL(1)分析\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout<<"本分析文法产生式为"<<endl;

for(int j=0;j<10;j++)

cout<<Production[j]<<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*LL(1)分析表\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout<<"\t+\t-\t\*\t/\t(\t)\ti\t#"<<endl;

cout<<"E\t\t\t\t\tE->TG\t\tE->TG"<<endl;

cout<<"T\t\t\t\t\tT->FS\t\tT->FS"<<endl;

cout<<"G\tG->+TG\tG->-TG\t\t\t\tG->ε\t\tG->ε"<<endl;

cout<<"F\t\t\t\t\tF->(E)\t\tF->i"<<endl;

cout<<"S\tS->ε\tS->ε\tS->\*FS\tS->/FS\t\tS->ε\t\tS->ε"<<endl;

string Sstring;

char T;

do{

cout<<"输入字符串"<<endl;

cin>>Sstring;//输入要分析的字符串

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*现进行如下分析\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout<<"步骤"<<"\t"<<"分析栈"<<"\t"<<"剩余输入串"<<"\t"<<"所用产生式"<<"\t"<<"动作"<<endl;

Analyse(Sstring);

count=0;//记录当前进行处理的输入字符串字符位置

line=0;//记录处理的步骤数

stacktd="#E";

while(!Stack.empty()){

cout<<Stack.top();

Stack.pop();

}

cout<<"是否继续分析,Y或y继续"<<endl;

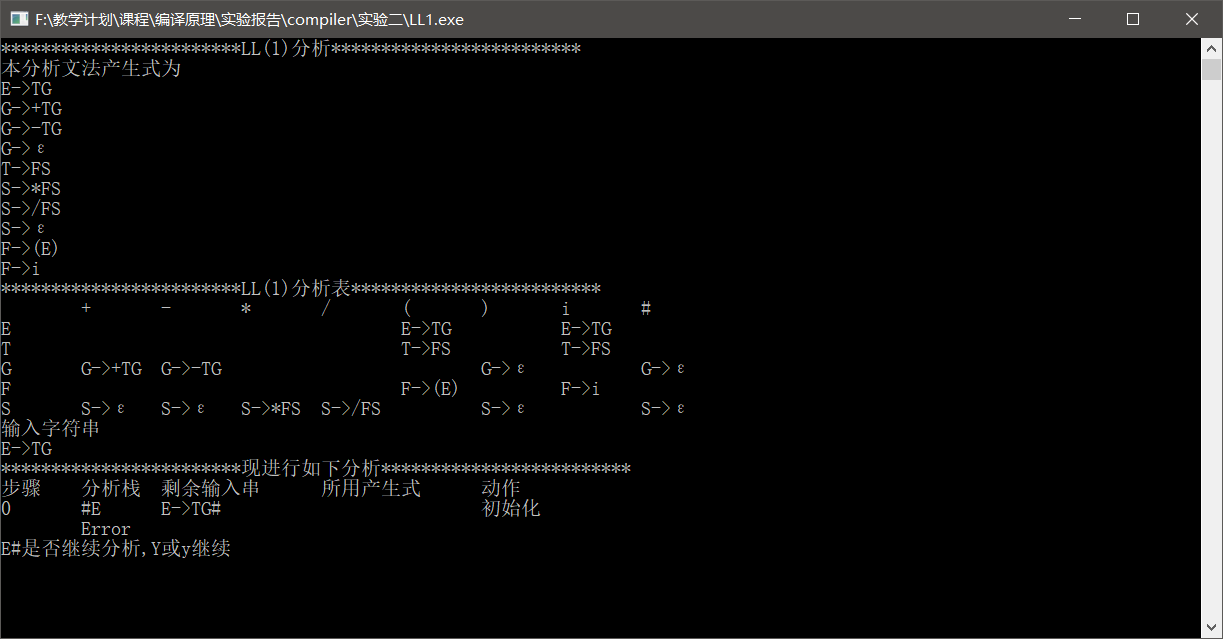
cin>>T;

}while(T=='y'||T=='Y');

return 0;

}

运行结果：



1. 实验结果的分析与评价（该部分如不够填写，请另加附页）

通过实验更好的了解了语言运行的根本，有利于我们以后更好的进行相关方面的学习。

注：实验成绩等级分为（90－100分）优，（80－89分）良，(70-79分)中，（60－69分）及格，（59分）不及格。